第1章

计算机抽象与技术

付俊宁 吕卫 译

计算机产业革命

- 计算机技术的进步
  - 摩尔定律的推动
- 让幻想触手可及
  - 车载计算机
  - 手机
  - 人类基因项目
  - 万维网
  - 搜索引擎
- 计算机无处不在

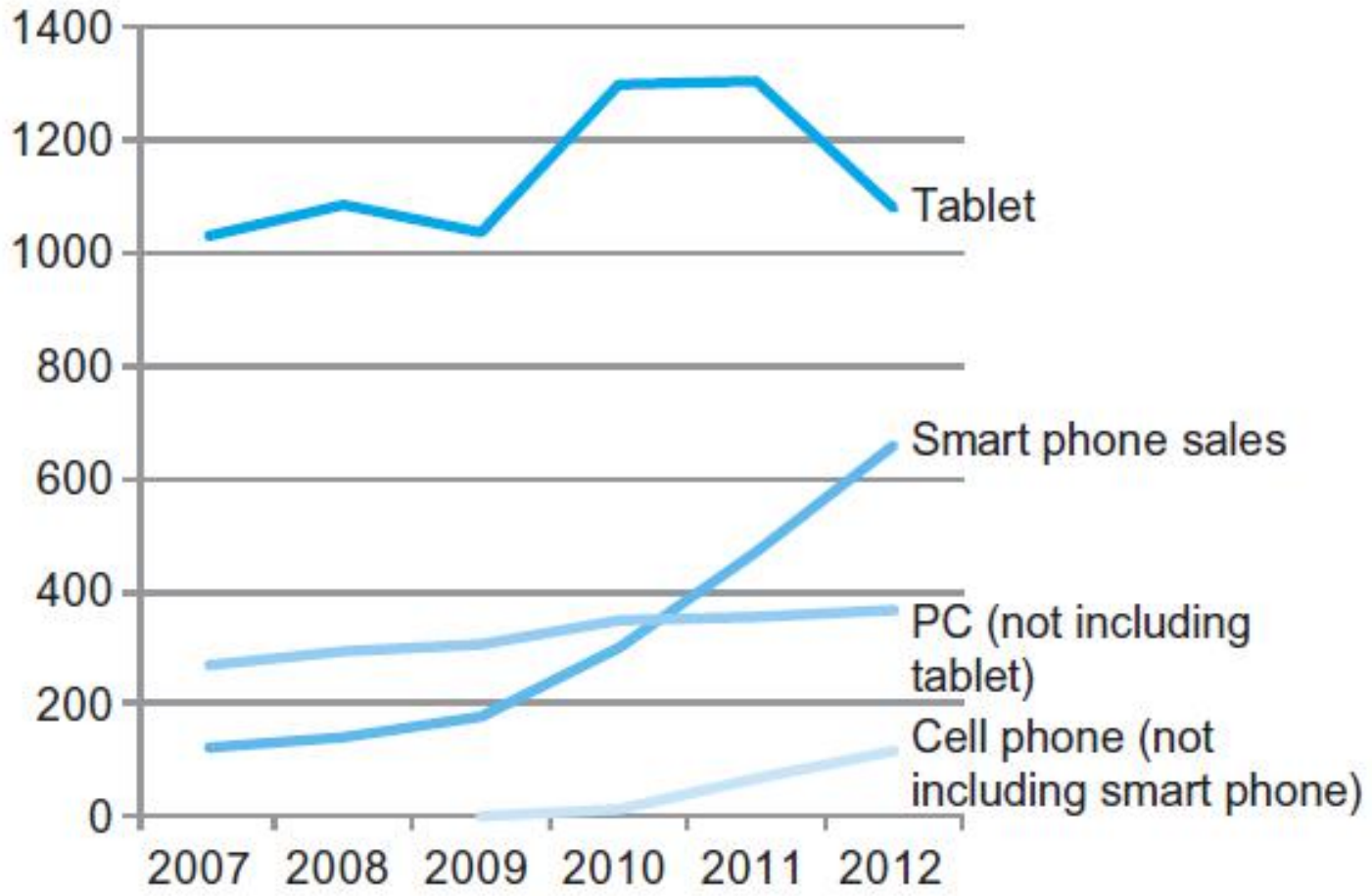
计算机的种类

- 个人计算机(PC)
  - 通用，各类软件
  - 受制于性价比
- 服务器
  - 基于网络
  - 大容量，高性能，高可靠性
  - 尺寸从小型服务器到建筑物大小的服务器

计算机的分类

- 超级计算机
  - 高端科学和工程计算
  - 计算能力最强，但在整个计算机市场所占比例很小
- 嵌入式计算机
  - 作为系统组件隐藏着
  - 有严格的功耗、性能、成本限制

后PC时代


1400
1200
1000
800
600
400
200
0
2007
2008
2009
2010
2011
2012
Tablet
Smart phone sales
PC (not including tablet)
Cell phone (not including smart phone)

后PC时代

- 个人移动设备(PMD)
  - 电池供电
  - 连接到Internet
  - 价格为几百美元
  - 智能手机、平板电脑、电子眼镜
- 云计算
  - 仓储规模计算机(WSC)
  - 软件即服务(SaaS)
  - 在PMD和云上各运行一部分软件
  - Amazon和Google

你将学到什么

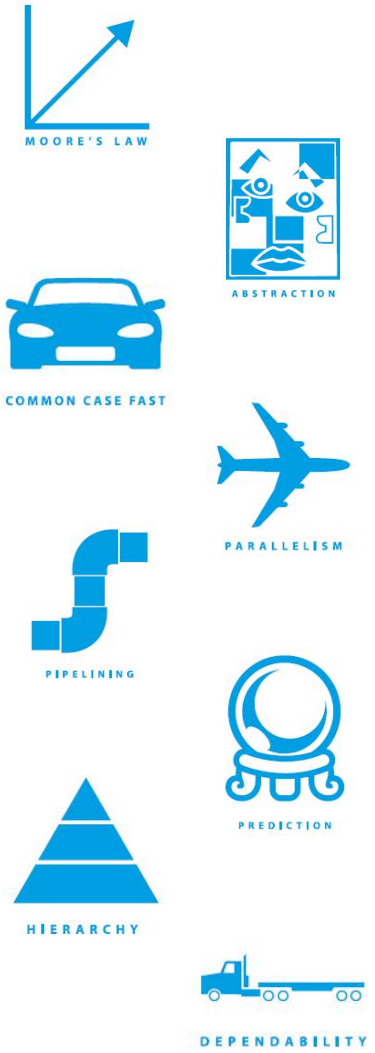
- 程序怎样被翻译成机器语言
  - 以及硬件如何执行程序
- 软硬件之间的接口
- 哪些因素决定了程序的性能
  - 以及如何提高程序性能
- 硬件设计者如何提高性能
- 什么是并行处理

理解程序性能

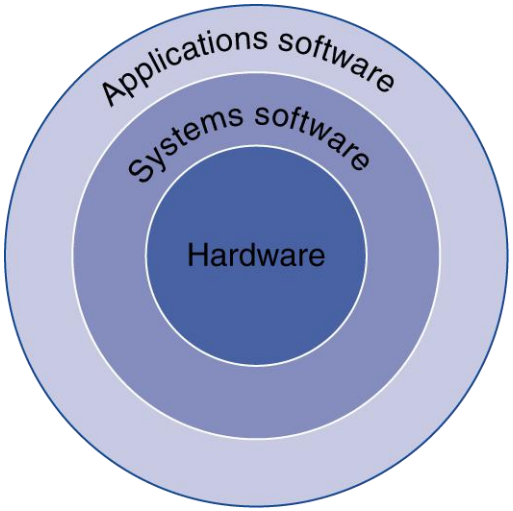
- 算法
  - 决定了执行的操作数量
- 编程语言、编译器和体系结构
  - 决定了每个操作对应的机器指令数量
- 处理器和存储系统
  - 决定了指令的执行速度
- I/O系统（含操作系统）
  - 决定了I/O操作的执行速度

8个伟大思想

- 面向**摩尔定律**设计
- 使用**抽象**简化设计
- **加速大概率事件**
- 通过**并行**提高性能
- 通过**流水线**提高性能
- 通过**预测**提高性能
- 存储器**层次**
- 通过冗余提高**可靠性**



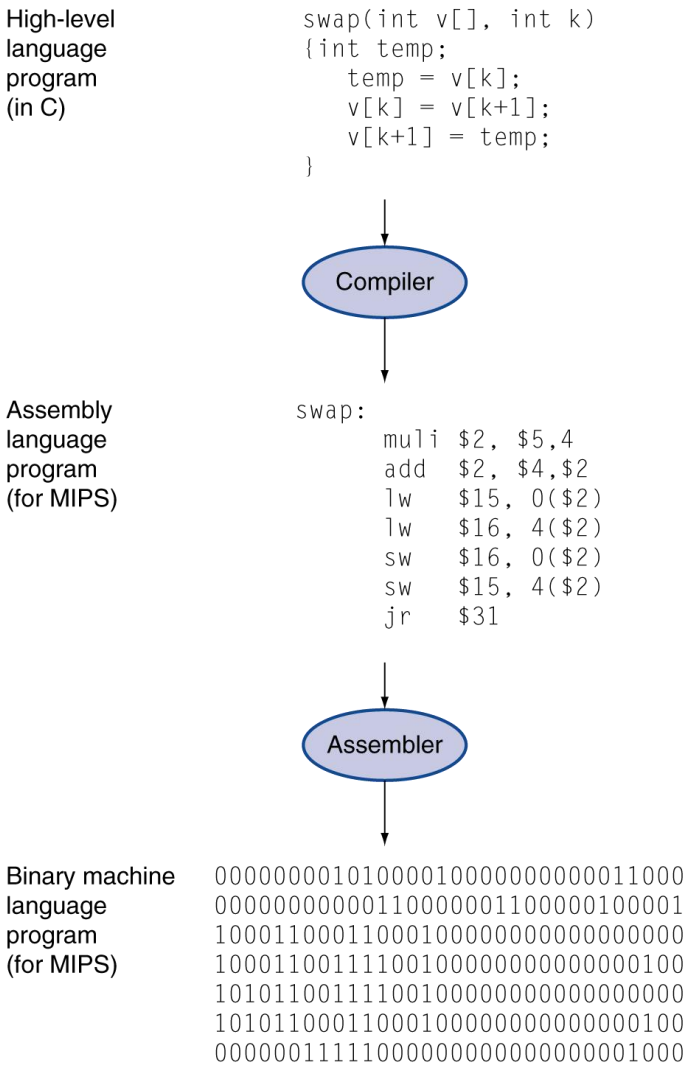
程序概念入门



- 应用软件
  - 用高级语言编写
- 系统软件
  - 编译器：把高级语言代码翻译为机器代码
  - 操作系统：服务代码
    - 处理输入/输出
    - 管理内存和外存
    - 规划任务与共享资源
- 硬件
  - 处理器、内存、I/O控制器

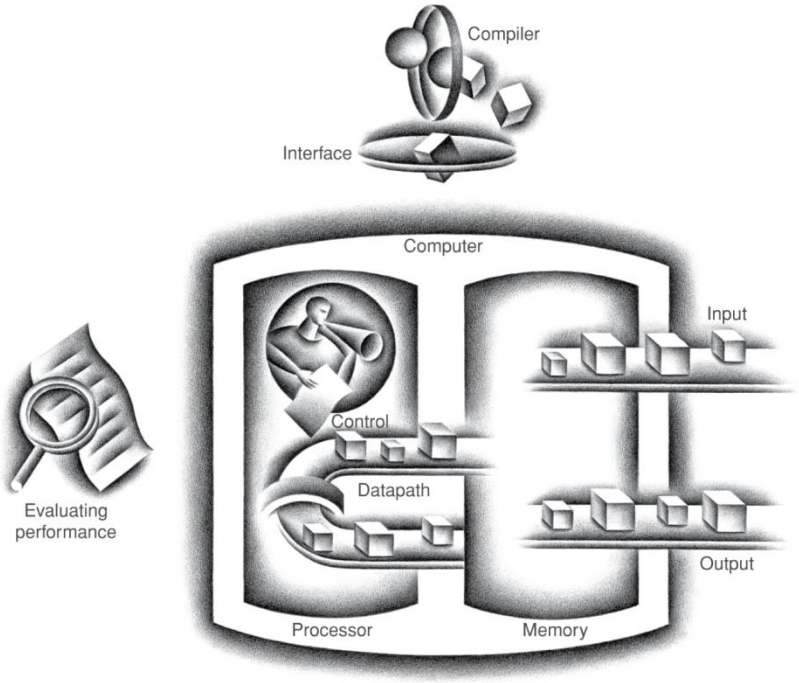
程序代码的层次

- 高级语言
  - 更接近问题域的抽象层次
  - 用于提高生产效率和可移植性
- 汇编语言
  - 机器指令的文本表示
- 硬件表示
  - 二进制数字（比特）
  - 编码的指令和数据



计算机的组成

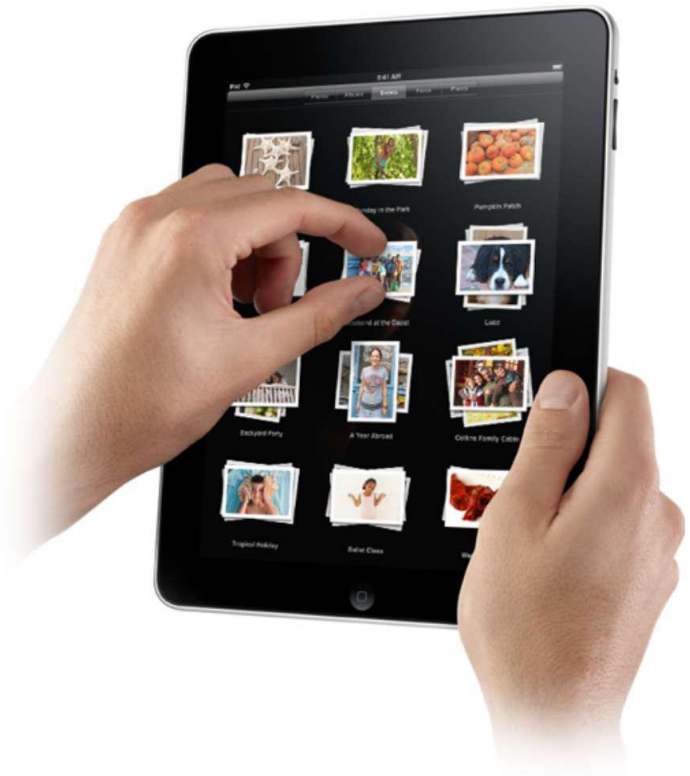
重点



- 所有类型的计算机有同样的组成
  - 台式机、服务器、嵌入式
- 输入/输出包括
  - 用户界面设备
    - 显示器、键盘、鼠标
  - 存储设备
    - 硬盘、CD/DVD、闪存
  - 网络适配器
    - 用来与其他计算机通信

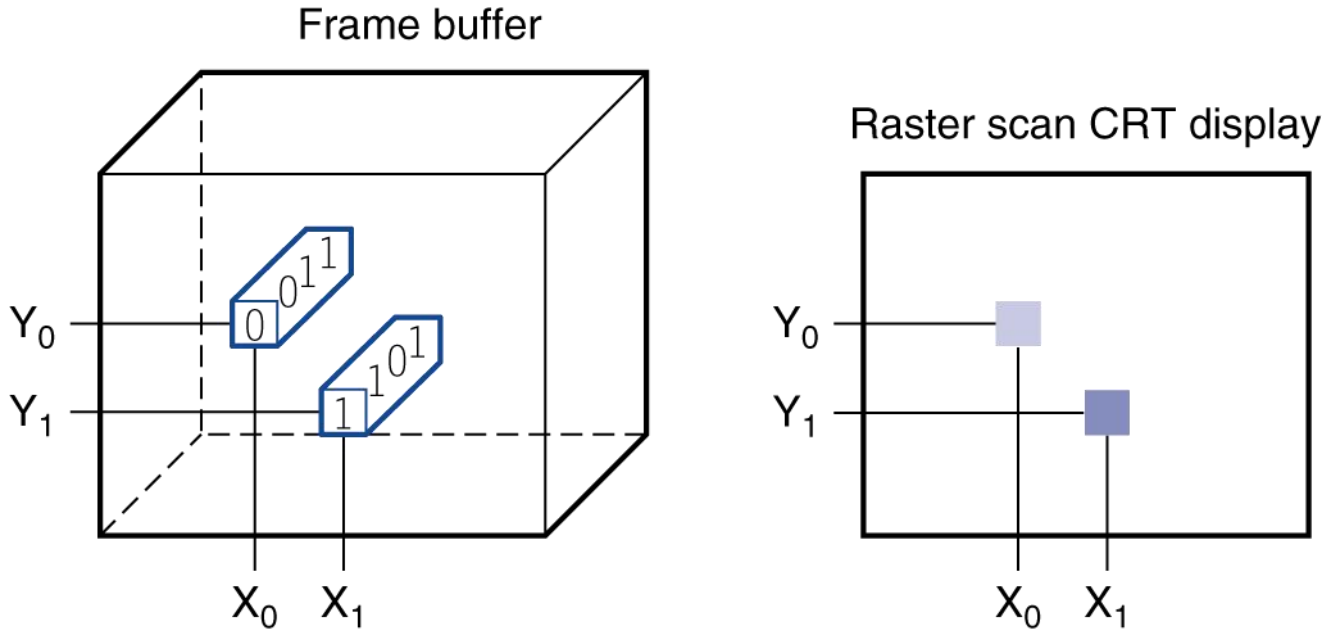
触摸屏

- 后PC设备
- 替代了键盘和鼠标
- 电阻式和电容式
  - 多数平板电脑、智能手机使用电容式屏幕
  - 电容屏支持多点触控

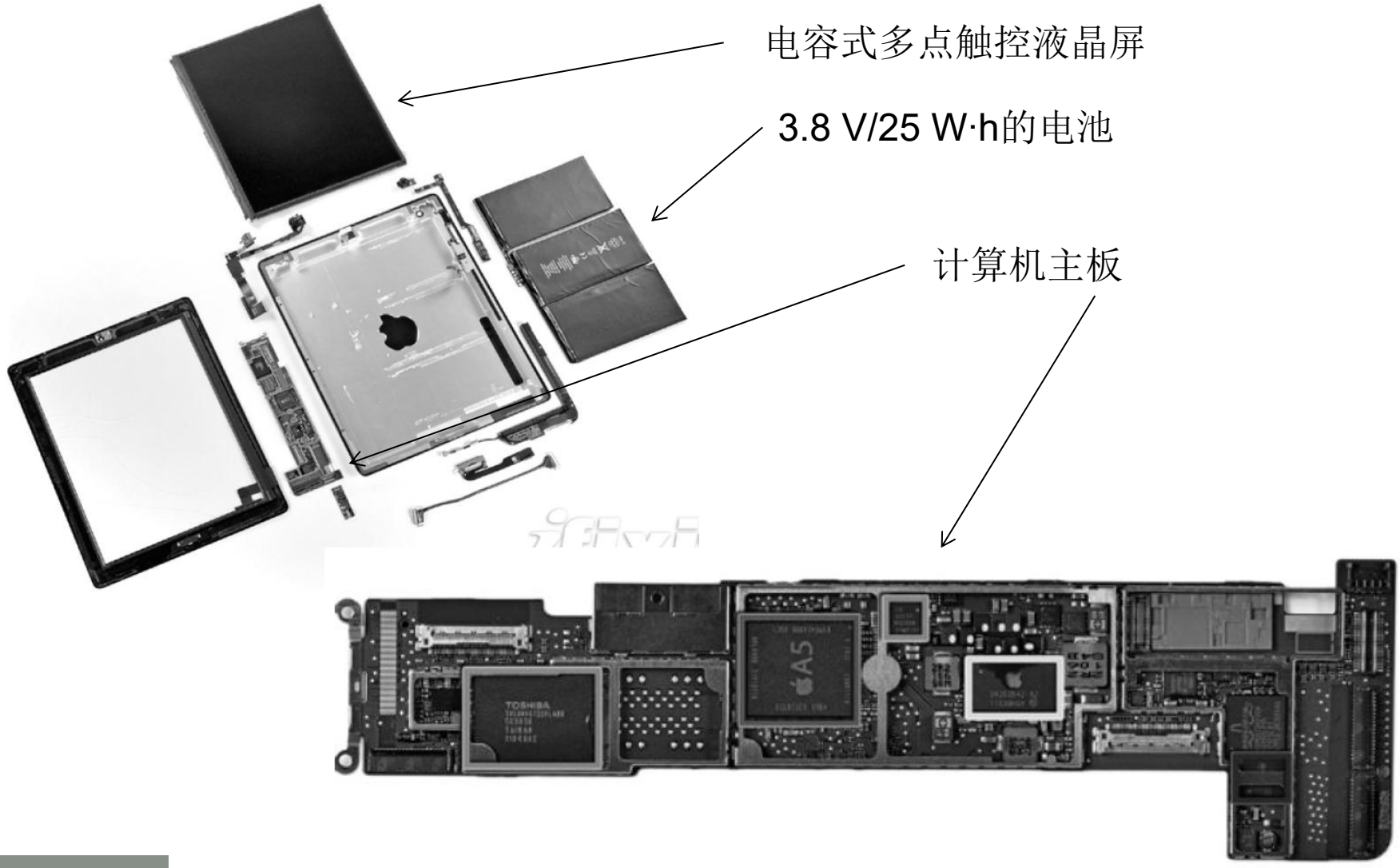


透过玻璃

- 液晶显示屏：图像元素（像素）
  - 帧缓存内容的映像



打开机壳

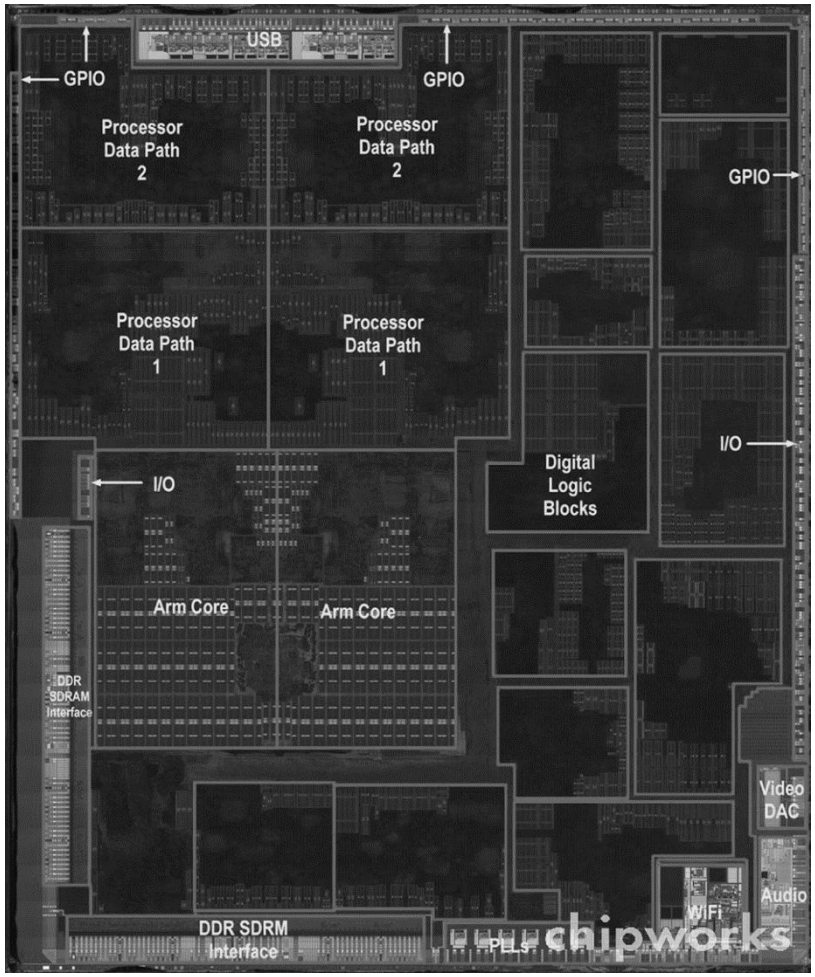


处理器(CPU)的内部

- 数据通路：对数据执行操作
- 控制器：排列数据通路、存储器等
- 高速缓存
  - 小而快的静态随机访问存储器，用于即时访问数据

处理器的内部

- Apple A5



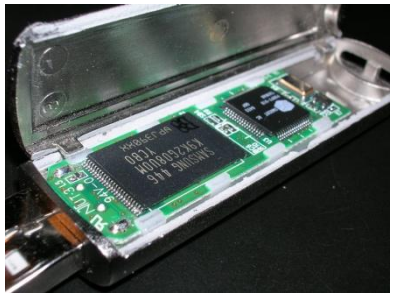
抽象

重点

- 抽象有助于我们处理复杂问题
  - 隐藏底层细节
- 指令集体系结构(ISA)
  - 硬件和软件的接口
- 应用二进制接口
  - ISA加上操作系统接口
- 实现
  - 底层细节和接口

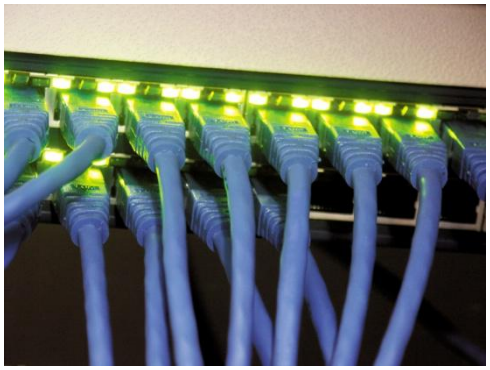
数据的安全场所

- 易失性主存储器
  - 断电时指令和数据将丢失
- 非易失性次级存储器
  - 磁盘
  - 闪存
  - 光盘(CDROM、DVD)



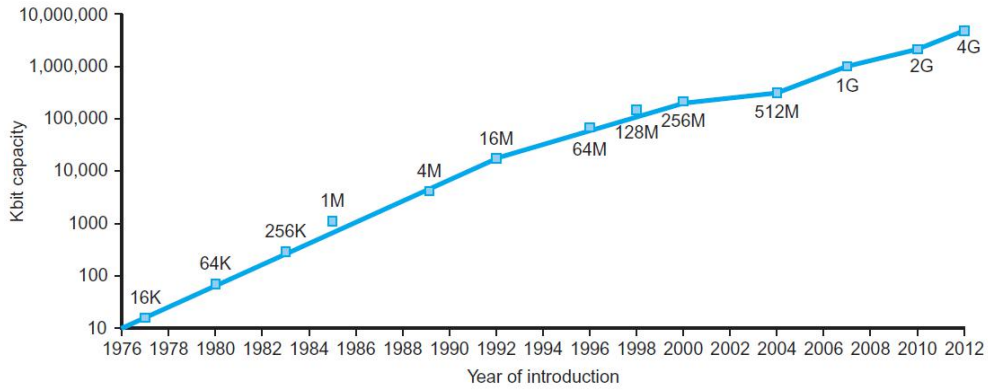
网络

- 通信、资源共享、非本地访问
- 局域网(LAN)：以太网
- 广域网(WAN)：互联网
- 无线网络：WiFi、蓝牙



技术趋势

- 电子技术不断演进
  - 容量和性能增加
  - 成本降低



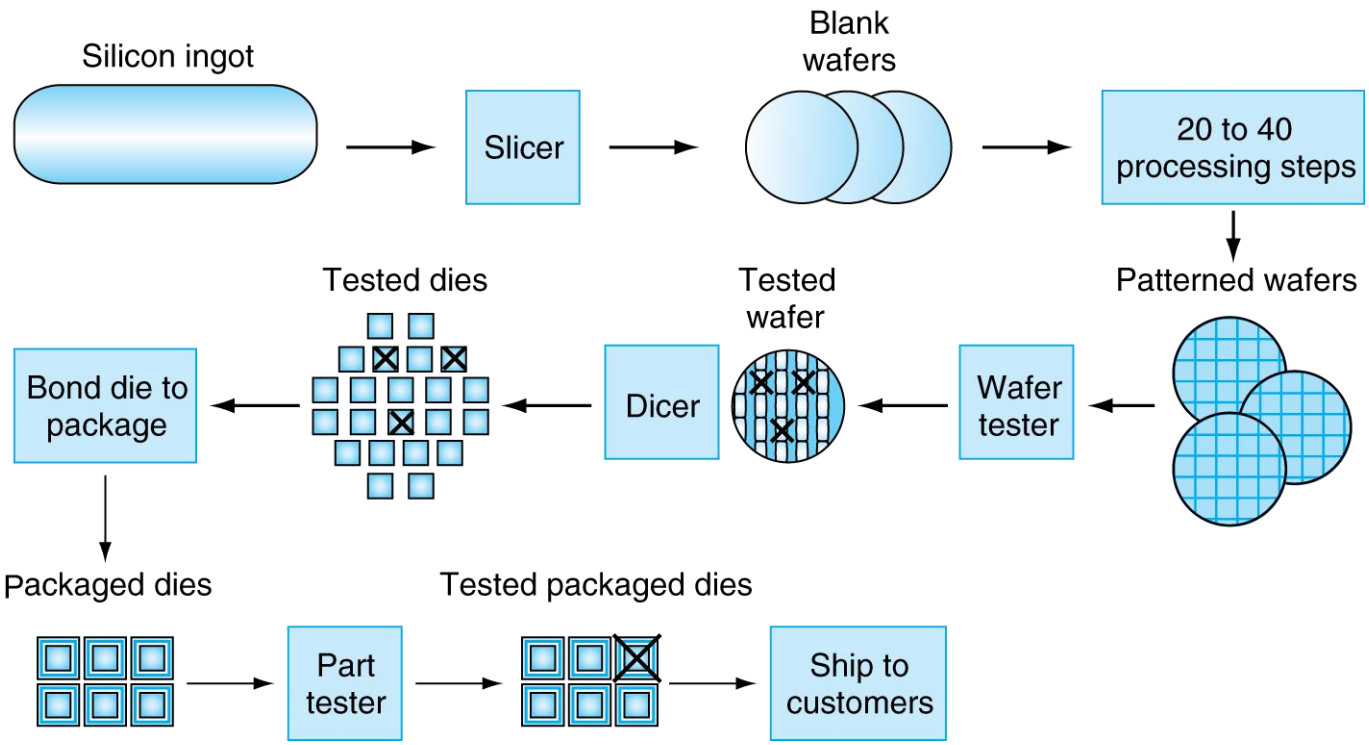
DRAM 容量

年代	技术	相对性价比
1951	真空管	1
1965	晶体管	35
1975	集成电路(IC)	900
1995	超大规模集成电路(VLSI)	2,400,000
2013	甚大规模集成电路	250,000,000,000

半导体工艺

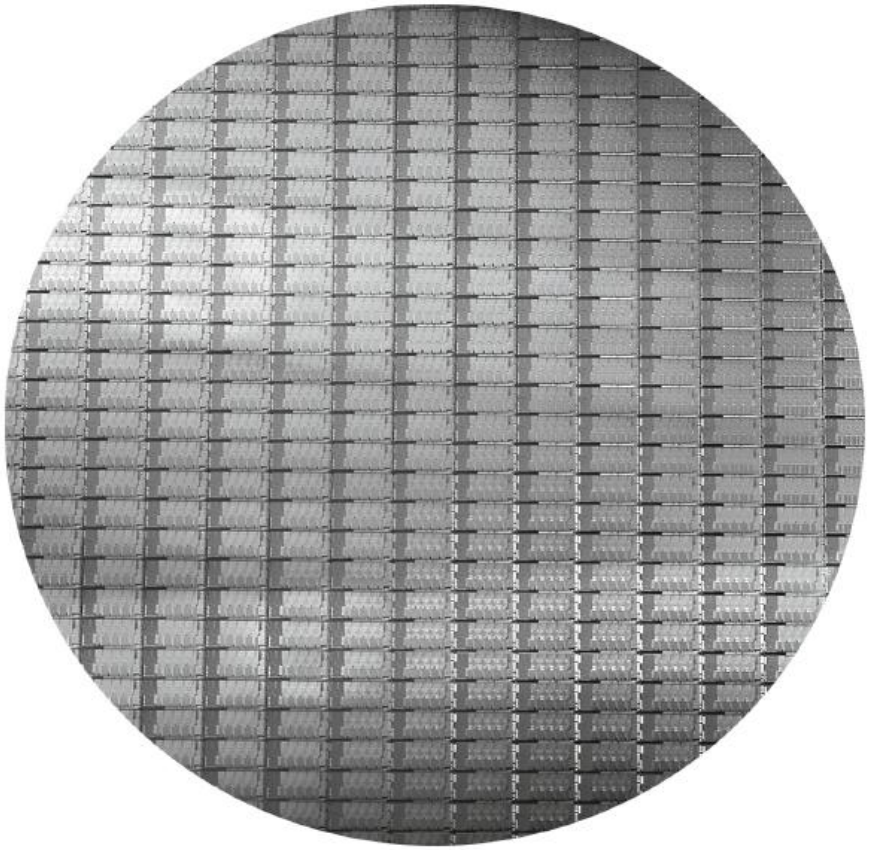
- 硅：半导体
- 添加材料以转变特性
  - 导体
  - 绝缘体
  - 开关

芯片的制造



- 成品率：每个晶圆产出合格芯片（裸片）的比例

Intel Core i7晶圆



- 300mm晶圆，280个芯片，32nm工艺
- 每个芯片是20.7 mm x 10.5 mm

集成电路的成本

$$\text{每个芯片的成本} = \frac{\text{每个晶圆的成本}}{\text{每个晶圆的芯片数} \times \text{成品率}}$$

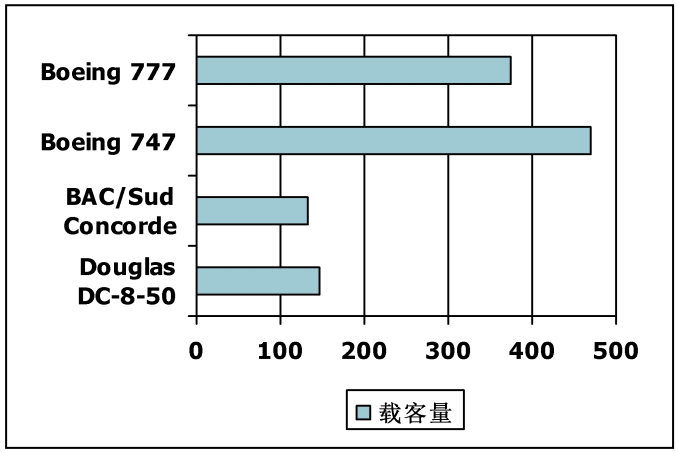
$$\text{每个晶圆的芯片数} \approx \text{晶圆面积}/\text{芯片面积}$$

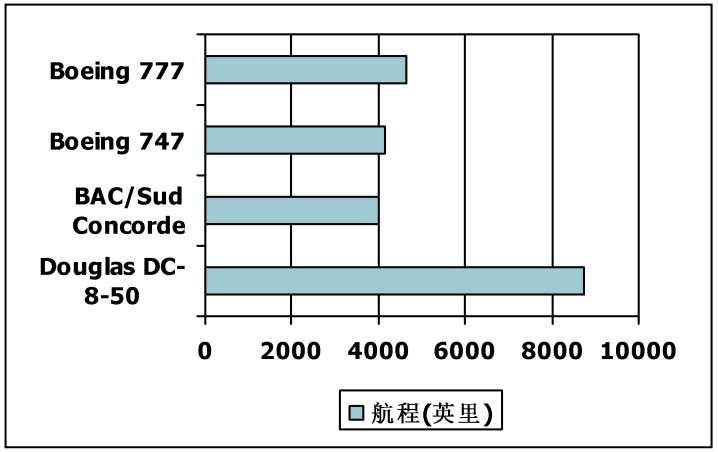
$$\text{成品率} = \frac{1}{(1 + (\text{单位面积的瑕疵数} \times \text{芯片面积}/2))^2}$$

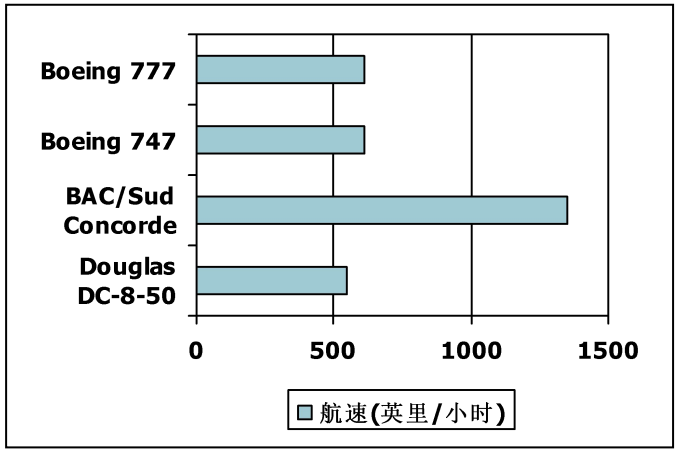
- 与面积和瑕疵率呈非线性关系
  - 晶圆的成本和面积是固定的
  - 瑕疵率取决于制造流程
  - 芯片面积取决于架构和电路设计

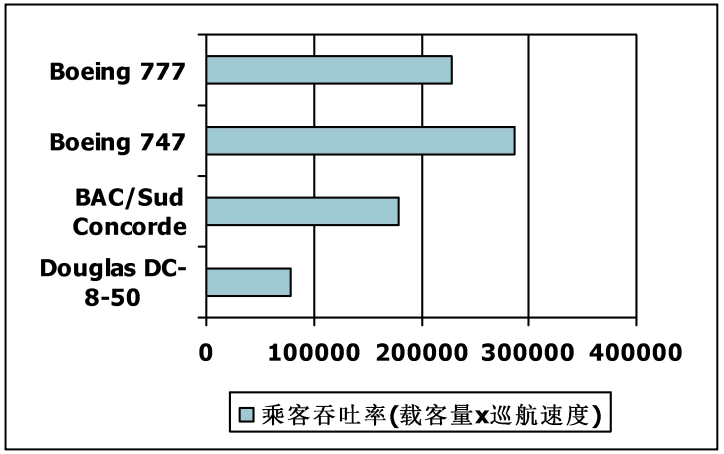
性能的定义

- 哪架客机的性能最好？









响应时间和吞吐率

- 响应时间
  - 完成一项任务所需时间
- 吞吐率
  - 单位时间内完成的任务数量
    - 例如每小时完成的任务、交易等等
- 在下列情况中，响应时间和吞吐率怎样变化？
  - 将处理器更换为更高速的型号
  - 添加更多处理器
- 我们先考虑响应时间…

相对性能

- 定义性能 = 1/执行时间
- “X是Y的n倍快”

$$\text{性能}_X / \text{性能}_Y = \text{执行时间}_Y / \text{执行时间}_X = n$$

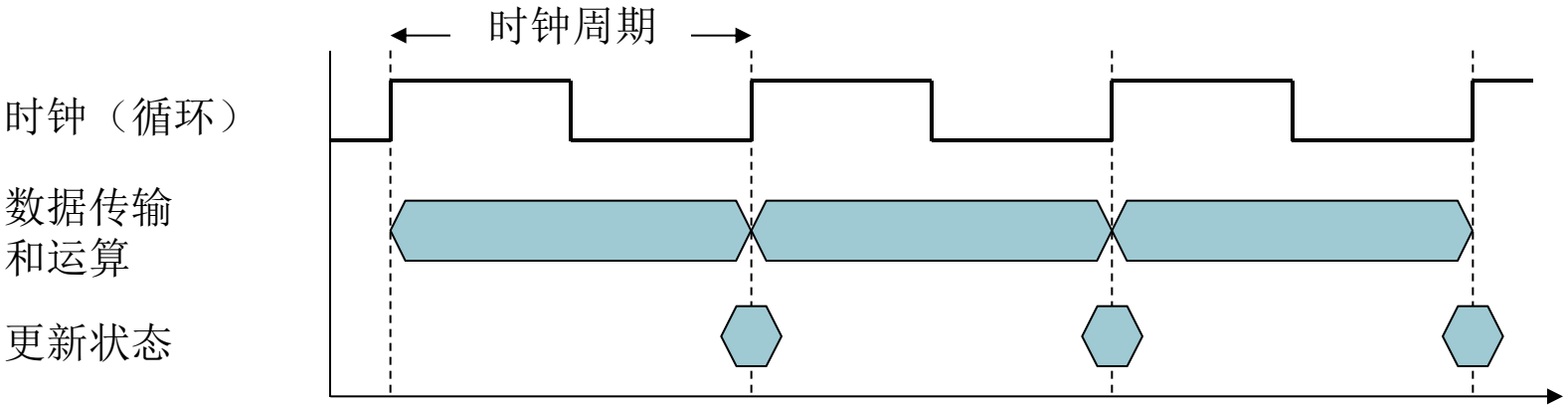
- 例：运行一个程序的耗时
  - A用10秒，B用15秒
  - 执行时间$_B$/执行时间$_A$ = 15秒/10秒=1.5
  - 所以A是B的1.5倍快

响应时间的度量

- 消逝时间
  - 总响应时间，包含所有方面
    - 处理、I/O、操作系统开销、空闲时间
  - 决定了系统性能
- CPU时间
  - 处理某一给定任务所花的时间
    - 不包括I/O时间、运行其他程序的时间
  - 包括用户CPU时间和系统CPU时间
  - 不同程序受CPU性能和系统性能的影响不同

CPU时钟

- 由一个恒定频率的时钟来掌控数字硬件的操作



  - 时钟周期：一个时钟循环的时间
    - 例如250ps = 0.25ns = 250×10^{-12}s
  - 时钟频率（速率）：每秒的周期数
    - 例如4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU时间

$$\text{CPU时间} = \text{CPU时钟周期数} \times \text{时钟周期时间}$$

$$= \frac{\text{CPU时钟周期数}}{\text{时钟频率}}$$

- 如何提升性能
  - 减少时钟周期数
  - 提高时钟频率
  - 硬件设计者必须经常权衡时钟周期数与时钟频率

CPU时间的例子

- 计算机A：时钟频率为2GHz，CPU时间为10秒
- 设计计算机B
  - 达到6秒的CPU时间
  - 可提高时钟频率，但造成时钟周期数为原来的1.2倍
- 计算机B的时钟频率得多高？

$$\text{时钟频率}_B = \frac{\text{时钟周期数}_B}{\text{CPU时间}_B} = \frac{1.2 \times \text{时钟周期数}_A}{6\text{秒}}$$

$$\text{时钟周期数}_A = \text{CPU时间}_A \times \text{时钟频率}_A$$

$$= 10\text{秒} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{时钟频率}_B = \frac{1.2 \times 20 \times 10^9}{6\text{秒}} = \frac{24 \times 10^9}{6\text{秒}} = 4\text{GHz}$$

指令数和CPI

$$\text{时钟周期数} = \text{指令数} \times \text{CPI}$$

$$\text{CPU时间} = \text{指令数} \times \text{CPI} \times \text{时钟周期}$$

$$= \frac{\text{指令数} \times \text{CPI}}{\text{时钟频率}}$$

- 一个程序的指令数
  - 由程序、ISA和编译器决定
- 每条指令的平均时钟周期数
  - 由CPU硬件决定
  - 如果不同的指令有不同的CPI
    - 平均CPI受指令组合的影响

CPI的例子

- 计算机A：周期= 250ps，CPI = 2.0
- 计算机B：周期= 500ps，CPI = 1.2
- ISA相同
- 哪台更快？快多少？

$$\text{CPU时间}_A = \text{指令数} \times \text{CPI}_A \times \text{周期}_A$$

$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A更快…

$$\text{CPU时间}_B = \text{指令数} \times \text{CPI}_B \times \text{周期}_B$$

$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU时间}_B}{\text{CPU时间}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

…快这么多

细说CPI

- 如果不同类指令的周期数不同

$$\text{周期数} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{指令数}_i)$$

- 加权平均CPI

$$\text{CPI} = \frac{\text{周期数}}{\text{指令数}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{指令数}_i}{\text{指令数}}}_{\text{相对频率}} \right)$$

CPI的例子

- 有两个编译过的代码序列可选，都使用A、B、C三类指令

指令种类	A	B	C
每类指令的CPI	1	2	3
序列1的指令数	2	1	2
序列2的指令数	4	1	1

- 序列1：指令数= 5
  - 时钟周期数
    = 2×1 + 1×2 + 2×3
    = 10
  - 平均CPI = 10/5 = 2.0
- 序列2：指令数= 6
  - 时钟周期数
    = 4×1 + 1×2 + 1×3
    = 9
  - 平均CPI = 9/6 = 1.5

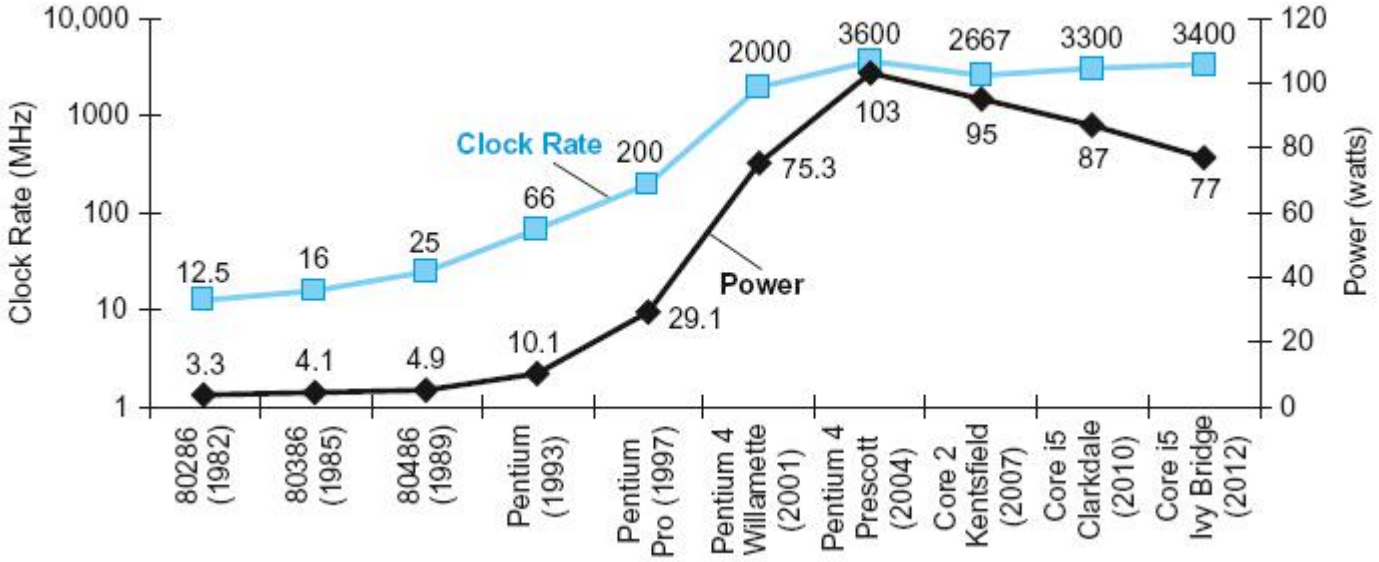
性能的总结

重点

$$\text{CPU时间} = \frac{\text{指令数}}{\text{程序}} \times \frac{\text{时钟周期数}}{\text{指令数}} \times \frac{\text{秒数}}{\text{时钟周期数}}$$

- 性能取决于
  - 算法：影响指令数，有可能影响CPI
  - 编程语言：影响指令数、CPI
  - 编译器：影响指令数、CPI
  - 指令集体系结构：影响指令数、CPI和时钟周期

功耗趋势



- CMOS集成电路技术

$$\text{功耗} = \text{容性负载} \times \text{电压}^2 \times \text{频率}$$

×30　　5V → 1V　　×1000

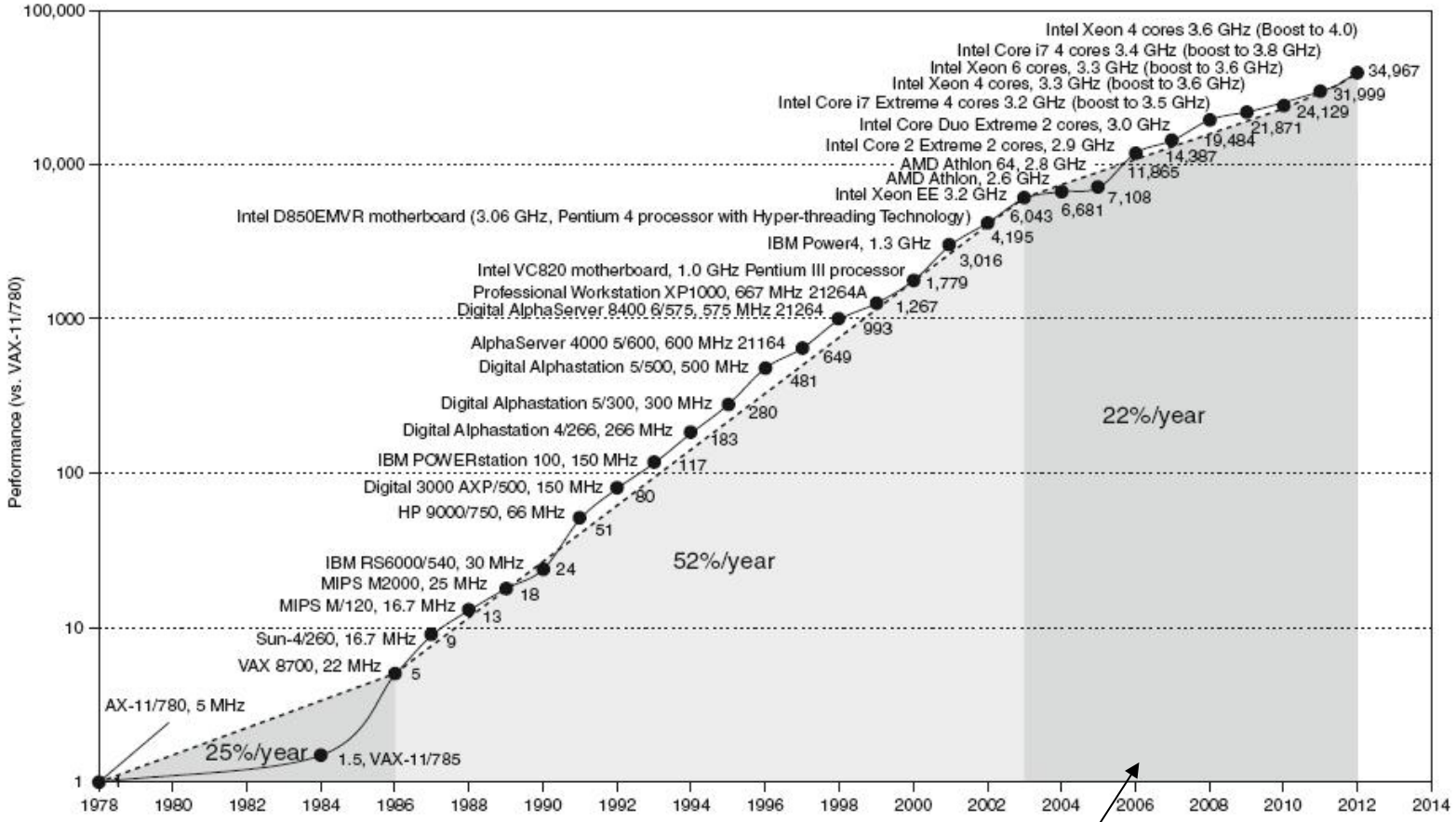
降低功耗

- 设想新CPU有着
  - 容性负载是旧CPU的 85%
  - 电压和频率都降低了15%

$$\frac{P_{新}}{P_{旧}} = \frac{C_{旧} \times 0.85 \times (V_{旧} \times 0.85)^2 \times F_{旧} \times 0.85}{C_{旧} \times V_{旧}^{\ 2} \times F_{旧}} = 0.85^4 = 0.52$$

- 功耗墙
  - 我们无法再降低电压
  - 我们无法消散更多热量
- 我们还能如何提高性能？

单处理器的性能



受到功耗、指令级并行度和存储器延时的限制

多处理器

- 多核微处理器
  - 每个芯片上有不止一个处理器
- 需要显式地并行编程
  - 与指令级并行相比
    - 硬件一次执行多条指令
    - 对程序员隐藏
  - 很难做到
    - 编程提高性能
    - 负载均衡
    - 优化通信和同步

SPEC CPU基准测试程序

- 用于衡量性能的程序
  - 当作典型的实际工作负载
- Standard Performance Evaluation Corp (SPEC)
  - 开发针对CPU、I/O、网络等等的基准测试程序
- SPEC CPU2006
  - 执行一组选定程序的消逝时间
    - I/O可忽略，所以着重考查CPU性能
  - 相对于参照计算机进行标准化
  - 归纳为性能比值的几何平均值
    - CINT2006（整型数）和CFP2006（浮点数）

$$\sqrt[n]{\prod_{i=1}^{n} \text{执行时间比}_i}$$

CINT2006在Intel Core i7 920上的运行结果

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

SPEC功耗基准测试程序

- 服务器在不同负载水平下的功耗
  - 性能：ssj_ops/秒
  - 功耗：瓦（焦/秒）

平均每产生1瓦的功耗完成的操作次数(Overall ssj_ops per Watt)

$$= \left(\sum_{i=0}^{10} \text{ssj\_ops}_i\right) \Big/ \left(\sum_{i=0}^{10} \text{功耗}_i\right)$$

SPECpower_ssj2008在Xeon X5650上的运行结果

Target Load %	Performance (ssj_ops)	Average Power (Watts)
100%	865,618	258
90%	786,688	242
80%	698,051	224
70%	607,826	204
60%	521,391	185
50%	436,757	170
40%	345,919	157
30%	262,071	146
20%	176,061	135
10%	86,784	121
0%	0	80
Overall Sum	4,787,166	1,922
Σssj_ops/Σpower =		2,490

陷阱：Amdahl定律

- 在改进计算机的某个方面时期望整体性能的提高与改进量成正比

$$\text{改进后的执行时间} = \frac{\text{受改进影响的执行时间}}{\text{改进量}} + \text{不受影响的执行时间}$$

- 例：乘法操作占100秒中的80秒
  - 如果要把该程序的运行速度提高到5倍，乘法操作的速度应该改进多少？

$$20 = \frac{80}{n} + 20$$

  - 做不到！
- 推论：加速大概率事件

谬误：空闲时功耗低

- 回顾i7的功耗基准测试
  - 100%负载时：258W
  - 50%负载时：170W(66%)
  - 10%负载时：121W(47%)
- Google数据中心
  - 多数时间在10% – 50%的负载下运行
  - 只有不到1%的时间达到100%负载
- 考虑设计功耗正比于负载的处理器

陷阱：用MIPS度量性能

- MIPS：Millions of Instructions Per Second（每秒百万条指令）
  - 没有考虑到
    - 计算机之间指令集体系结构的差异
    - 指令之间复杂程度的差异

$$\text{MIPS} = \frac{\text{指令数}}{\text{执行时间} \times 10^6} = \frac{\text{指令数}}{\frac{\text{指令数} \times \text{CPI}}{\text{时钟频率}} \times 10^6} = \frac{\text{时钟频率}}{\text{CPI} \times 10^6}$$

  - 对给定的CPU，CPI随程序变动

本章小结

- 性价比在提高
  - 得益于底层技术的进步
- 不同层次的抽象
  - 在硬件和软件中都存在
- 指令集体系结构
  - 硬件/软件接口
- 执行时间：最好的性能度量
- 功耗是一个限制因素
  - 用并行提高性能